**PROGETTO OOP a.a 2017-2018**

Gruppo Progetto : Di Egidio Laura 248831

Gentile Fabio 248809

1.1 Analisi dei requisiti

Funzionalità del sistema:

Il nostro sistema deve gestire ed organizzare un insieme di collezioni, servizi e persone…

Successivamente i requisiti per ogni sottosistema che siamo riusciti a trovare sono i seguenti:

1 VIEWER

* 1. Login : accesso alla consultazione delle opere digitali ad utenti registrati.
  2. Ricerca : Un utente può svolgere la ricerca nel catalogo per metadati (anno, titolo..) oppure all’ interno del testo della trascrizione.
  3. Scelta dell’ opera : appare a sx l’ immagine della pagina dell’ opera, a dx il testo trascritto (se disponibile). Le pagine vengono sfogliate attraverso un paginatore.
  4. Download dell’ opera : Operazione a disposizione solo per utenti speciale (immaginiamo ad esempio che l’ utente speciale è per noi un utente abbonato).
  5. Compilare il modulo : Un utente può diventare un collaboratore e trascrivere testi compilando un modulo.
  6. Dati Utente : L’utente può accedere al proprio profilo e visualizzare i dati della registrazione.

1. UPLOADER

2.1 Controllo digitale : La digitalizzazione dell’ immagine è controllata da supervisori per assicurarne la correttezza e la qualità.

1. TRANSCRIBER

3.1 Operazioni di trascrizione : Le immagini vengono trascritte manualmente attraverso in formato TEI (Text Encoding Initiative).

3.2 Modifiche : Devono essere sincronizzate in quanto più trascrittori possono lavorare alla stessa pagina

* 1. Revisione Trascrizioni : Le trascrizioni sono soggette a revisioni da parte di revisori.

1. MANAGER

4.1 Assegnazione immagine : Tale sottosistema gestisce le assegnazioni di una o più immagini ad uno o più trascrittori.

4.2 Revisiona/Corregge/Valuta/Riassegna : Il sottosistema revisiona, corregge o valuta le trascrizioni concluse. Può riassegnare pagine ai trascrittori.

4.3 Pubblicazione : Pubblicazione delle trascrizioni e delle opere.(IMMAGINI)

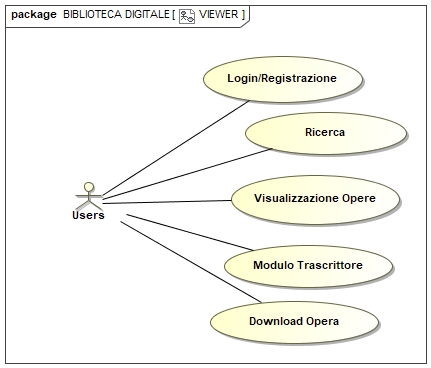
4.4 Livello Trascrittore: (1-5) Il sottosistema gestisce il livello dei trascrittori in base al livello d’esperienza.

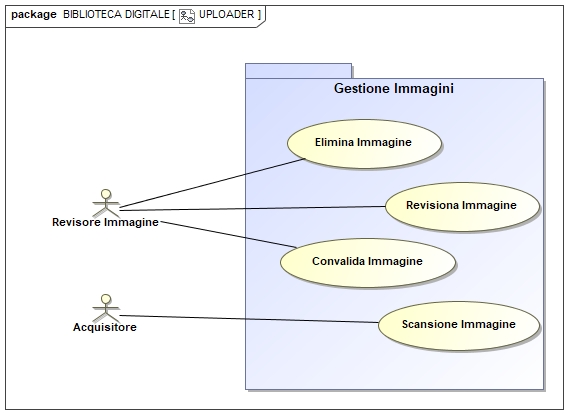
4.5 Supervisione acquisizione : Supervisione dell’ acquisizione delle immagini

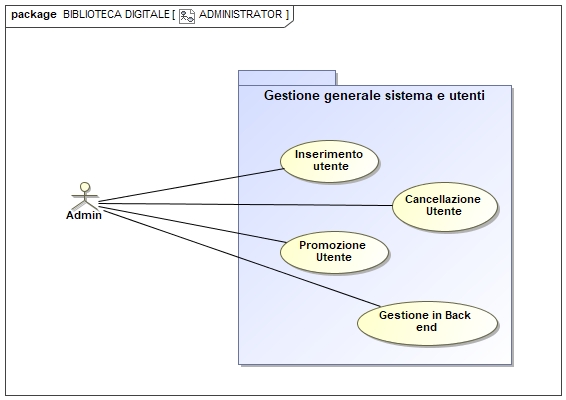
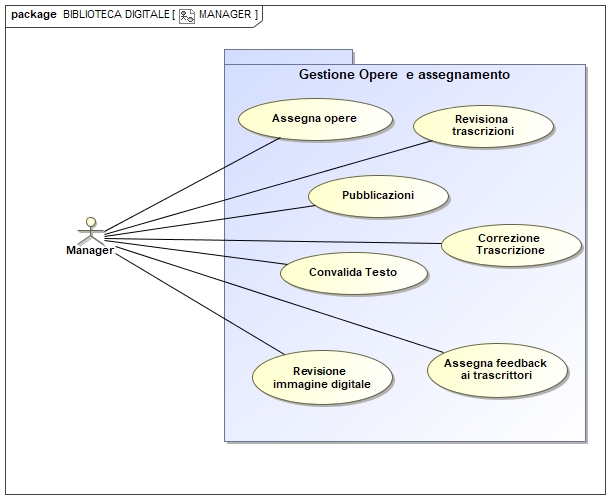
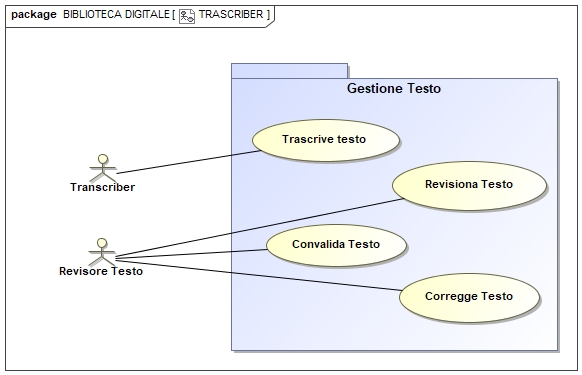
5 ADMINISTRATOR5.1 Gestione totale: Gestisce in back-end tutto il sistema (Anagrafica, opere, ecc..)

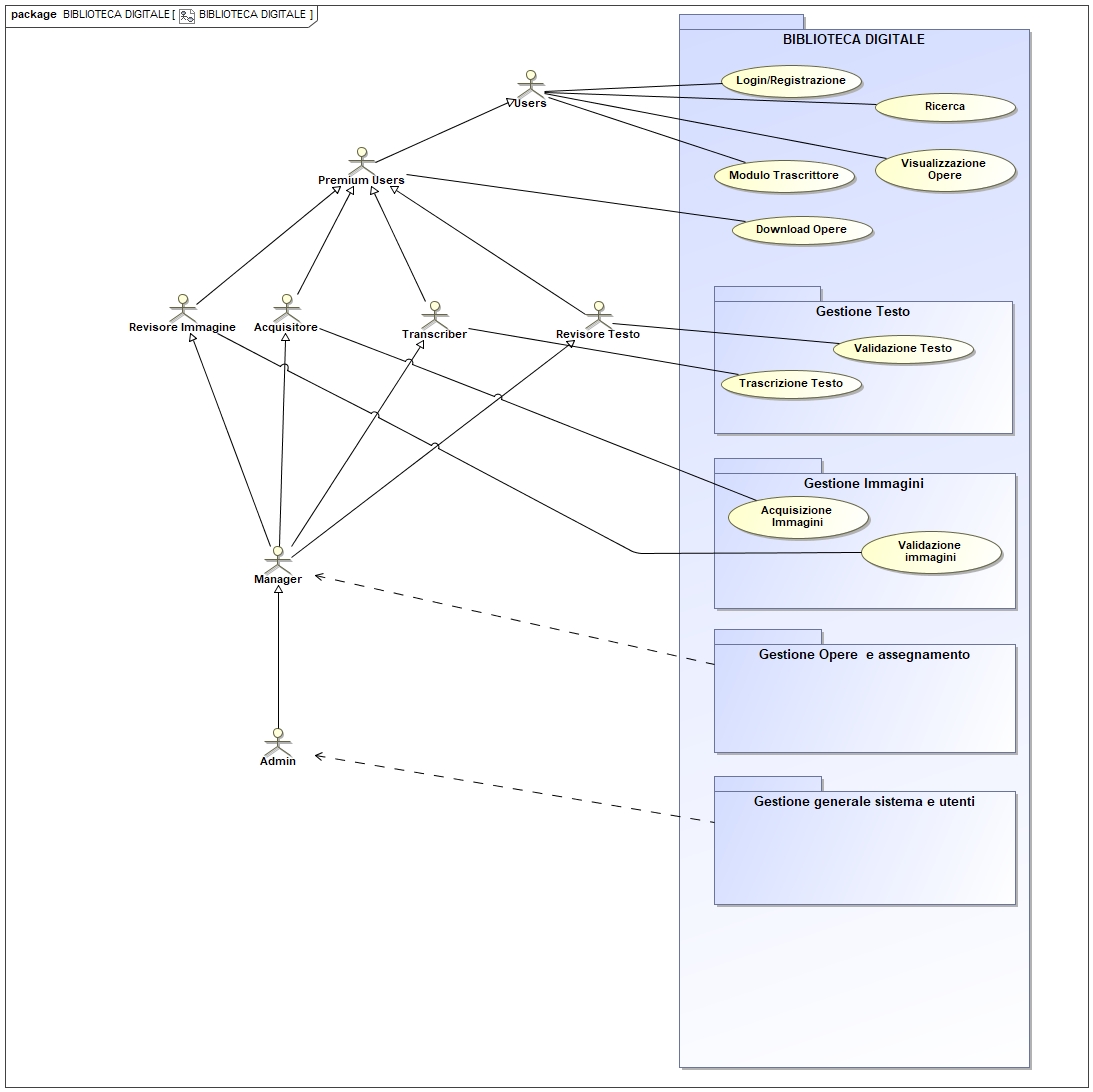
1.2 MODELLI UML USE CASE

Per definire al meglio lo Use Cases Diagram completo abbiamo di nuovo suddiviso gli attori con le loro funzionalità nei sottosistemi

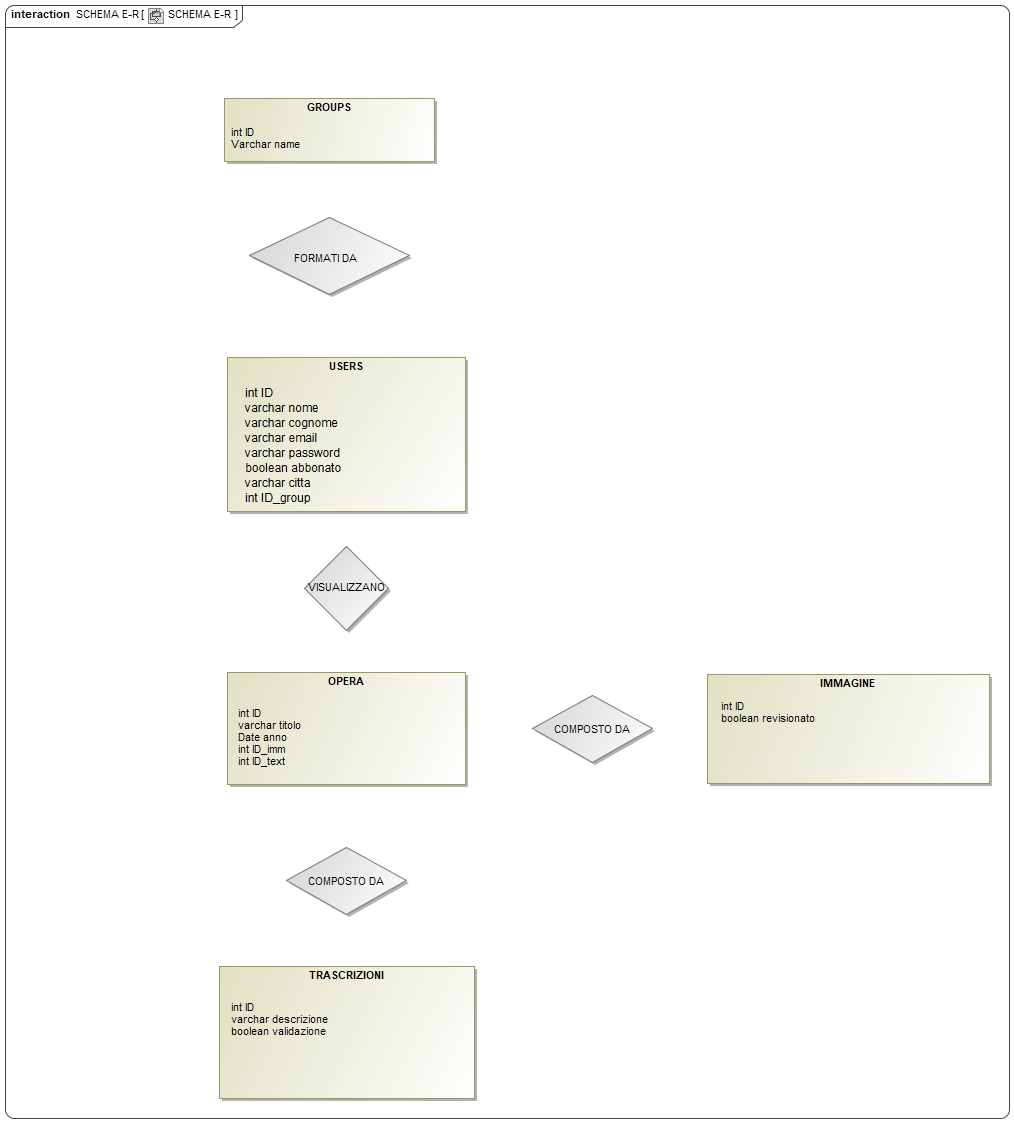






UML USE CASES COMPLETO

Prima di determinare il Modello di Dominio è stato utile, per una maggior visione delle entity, crearci lo schema ER.

****

(1,n)

(1,n)

(1,1)

(1,1)

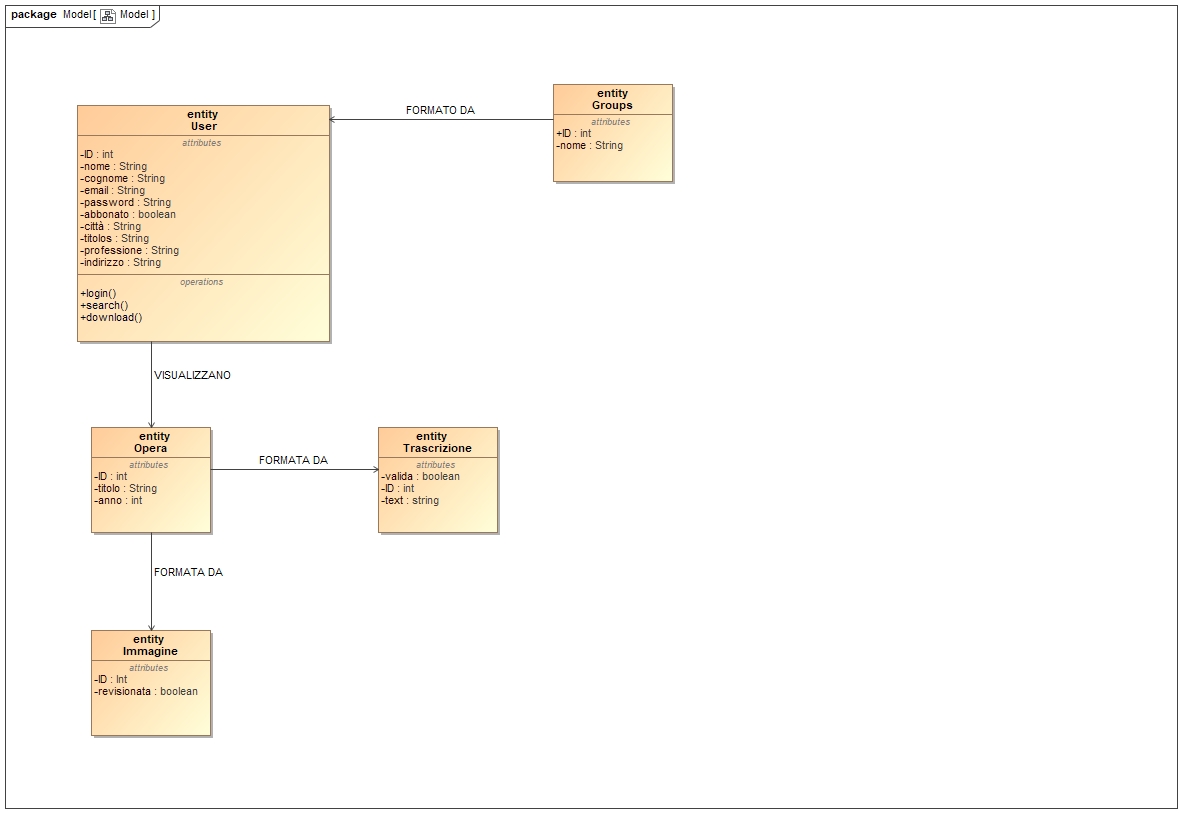
(1,n)

(1,n)

(1,1)

(1,n)

1.3 DOMAIN MODEL



1.4 *analisi finalizzata all’individuazione di classi entity, boundary*

*e controller*

ENTITY

|  |  |  |
| --- | --- | --- |
| ENTITY | ATTRIBUTI | DEFINIZIONE |
| Groups | ID  Nome | In base all’ ID e al nome del tipo ‘manager, trascrittore...’ capiamo quale ruolo svolgo il nostro utente all’interno del sistema. Tutti gli utenti sono per noi utenti registrati ma che in base al ruolo definito appunto da Groups siamo in grado di attribuirgli determinate azioni. |
| Users | ID  Nome  Cognome  Abbonato  Email  Password  Citta  Indirizzo  Titolo di studio | Nella classe Users l’utente può visualizzare il proprio profilo determinato dai seguenti attributi. Inoltre, l’utente può svolgere le azioni quali: login,  registrazione, modifica,  revisione, pubblicazione,  download, trascrizione,  acquisizione, feedback… |
| Opera | ID  Titolo  Anno | Opera formata da un codice ID, il titolo dell’opera e l’anno di pubblicazione. |
| Immagine | ID  Revisionato | Nell’immagine abbiamo l’attributo revisionato (boolean) da dove specifichiamo se la scansione di una certa opera è stata revisionata successivamente |
| Trascrizione | ID  Convalida  Text | Anche la trascrizione ha un attributo booleano ‘convalida’ che specifica se la trascrizione sia stata o meno convalidata. |

|  |  |
| --- | --- |
| CheckImage() | Utilizzato dal revisore per la convalida dell’immagine acquisita |
| ScanImage() | Scansione immagine da parte dell’ acquisitore |
| CheckText() | Validazione del testo digitale da parte del revisore testo |
| TrascrText() | Utilizzato per la trascrizione dal trascrittore |
| PubOp() | Pubblicazione dell’ opera da parte del manager e dal revisore dell’ immagine in caso non ci siano trascrizioni |
| Select\_Op() | Il manager decide di assegnare un opera al trascrittore |
| Modulo\_Online() | L’utente compilando un modulo online diventa un trascrittore delle opere |
| Revisiona() | Il manager revisiona l’ immagine digitale e il testo trascritto |
| ButtonLogin() | Pulsate per accedere al profilo dell ‘ Utente |
| ButtonDownload() | Pulsante per effetture il download dell’opera |

BOUNDARY

CONTROL OBJECT

Analizzando le funzionalità del sistema abbiamo individuato diversi controlli che il sistema dovrà fare all’ utente in diverse occasioni.   
ManagerControl() : Controllo utilizzato nel momento in cui un utente voglia modificare opere, assegnare trascrizioni, dare feedback ai trascrittori o pubblicare un’ opera.  
PremiumUsersControl() : Controllo per gli utenti che vogliono scaricare l’ opera. Visualizza se l’utente è un utente abbonato (Premium) o viceversa.  
UsersAcquisizioneControl() : Controlla se un utente può o meno acquisire un immagine.  
UsersTrascrControl(): Controlla se un utente può o meno trascrivere il testo dell’ opera.  
RevisoreControl(): Controlla se l’utente è un revisore. In questo caso verifica se è un revisore Trascrittore o revisore Acquisizione. Da ciò può dunque revisionare e convalidare l’immagine/testo.  
UsersRegisteredControl() : Controlla nel momento del login se l’utente è registrato.

2 SYSTEM DESIGN

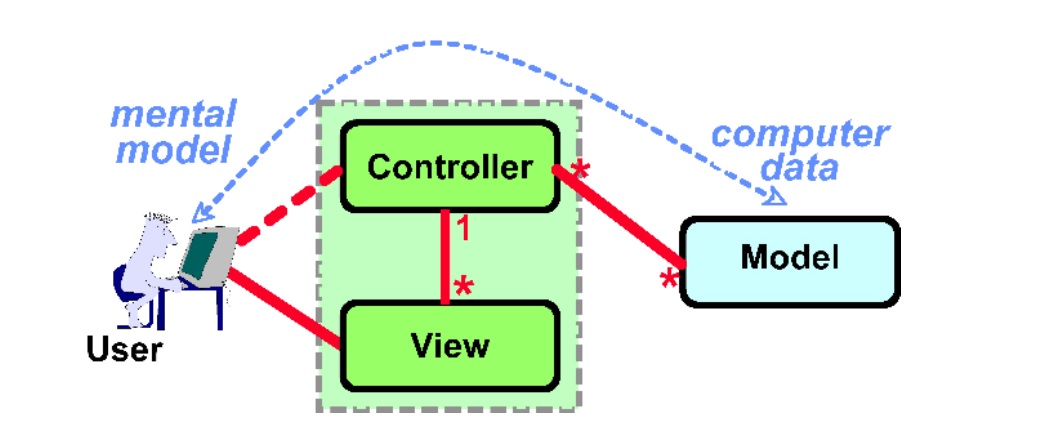
ARCHITETTURA SOFTWARE

È stato scelto dal nostro gruppo il pattern MVC (Model View Controller), in quanto permette una

suddivisione del sistema in parti indipendenti tra di loro più facili da gestire.

Il pattern architetturale è così diviso

* Rappresentazione del modello di dominio (Model)
* Interfaccia Utente (View)
* Controllo operazioni Utente-Sistema (Controller)



MVC è un pattern architetturale che ci permette di separare la logica di presentazione dei dati da quella di business.

I componenti software sono divisi in base ai compiti svolti all’interno del sistema

* Il Model fornisce i metodi per accedere ai dati utili. Gestisce direttamente i dati, la logica e le regole dell’applicazione.
* Il View si occupa della visualizzazione dei contenuti. È una qualsiasi rappresentazioni in output di informazioni, nel nostro esempio le immagini, il testo ecc..
* Il Controller riceve i comandi dall’ utente attraverso il view e li svolge alterando lo stato degli altri due componenti.

